Autonomous docking of a tracked wheels robot to its tether cable using a vision-based algorithm

Fausto Ferreira and Rodrigo Ventura

Abstract— **Search and Rescue (SAR) Robotics has been gaining an increasing interest in recent years. In spite of that, there are still many challenges to be addressed in terms of autonomy level. RAPOSA is a semi-autonomous, tracked wheels robot, designed for SAR operations. This paper presents an autonomous docking algorithm. The docking is performed to a docking station that provides both power and a wireless access point. The docking station consists on a small, lightweight pyramidal structure at the end of a flexible cable, allowing the robot to continue operating while attached to this cable. The autonomous docking is mainly based on both vision, whenever possible, and dead reckoning (odometry) that was also developed specially for this work. The results show the effectiveness of the proposed algorithm, even in the presence of difficult environmental conditions.**

Keywords: **Search and Rescue Robotics, Autonomous Docking, Vision based control**

I. INTRODUCTION

In scenarios of natural catastrophes, a quick response by rescue teams is essential, but not always possible due to the hazards to human teams (e.g., buildings near collapse). Search and Rescue (SAR) robots able to operate in these contexts can act rapidly [1], for instance to gain situation awareness, and to provide voice contact with victims.

RAPOSA is one example of a semi-autonomous robot designed for these situations [2]. The main features of this robot are motor articulation between its main body and a frontal arm, the two-side tracked wheels, the wireless communications with or without the cable (access point at the end of the cable), and the possibility of operating the robot even if it flips down. The tethered option includes both wireless communication and power. The robot has three vision cameras: two in the frontal arm, pointing forward in the direction of the body, and one in the back, to aid docking operations.

This paper is organized as follows. In the next section the odometry is discussed. Then, the related work on autonomous docking is reviewed. The description of the autonomous docking is presented next, followed by the experimental results. This paper closes with some conclusions and future work.

II. AUTONOMOUS DOCKING

Several methods for automatic docking can be found in the literature. The use of computer vision has some advantages: on the one hand, it is a passive sensor eliminating the need of an emitting source, and on the other hand, other sensors like ultrasonic ones or infrared are not as accurate. In this work, only one camera is used to provide feedback to the control algorithm. Color segmentation, together with some geometric considerations, are used to perform the docking.

A. Related work

The first successful attempts to an autonomous mobile robot recharge were in the late 40's with the *Elsie* and *Elmer* the so called "Machina Speculatrix" by Walter [3]. These turtle-shaped robots had a light following behavior into a hut that contained the battery recharger and a light bulb. In the last decade, there were several docking strategies for recharging purposes, as well as other purposes. In [4] a Nomad XR4000 used a IR beacon, a sonar, and a Sick laser range finder to dock to a recharging station. More recently, a Radio-Frequency IDentification (RFID) docking was presented in [5]. Among the ones that use computer vision, in [6] it is described a docking system for marsupial robots with an imprinting biological inspired mechanism. Vision and a laser range finder were used in [7] for a Pioner 2DX. In [8], the Nomad has an omnidirectional camera and uses 3 non-collinear landmarks and their bearings to drive autonomously to the docking station. In [9], a circle on a wall is used to aid the docking. Another example of a docking purely relying on vision is described in [10] but it involves comparing features to a model stored in database.

B. Docking system

The docking system is composed by two parts: the grabbing mechanism (inside the robot) and the cable. The cable is flexible but at the end it has a solid pyramidal structure designed to provide a fixed inclination of it while laying on the ground (Fig. 1). To perform the docking the robot should lower its back and enter the metal guide with a certain orientation. Fig. 2 shows the docking socket at RAPOSA's back, with the camera visible inside the robot. In its back part, there are two sliding doors that grab and pull the pyramid into the robot (an effect of bi-conical shape at the end). The effectiveness of this solution from the design point of view was address before in [11]. Nevertheless, it is important to note that this system allows real-time operation because it provides power and communications from the docking station.

Comparing with the state-of-the-art, with the exception of [6], most of the publications on this subject use a visual target as an auxiliary tool to make the docking possible.

This work was partially supported by FCT (ISR/IST plurianual funding) through the POS_Conhecimento Program that includes FEDER funds.

Fausto Ferreira and Rodrigo Ventura are with Institute for Systems and Robotics / Instituto Superior Técnico, 1049-001 Lisbon, Portugal {fferreira,yoda}@isr.ist.utl.pt.

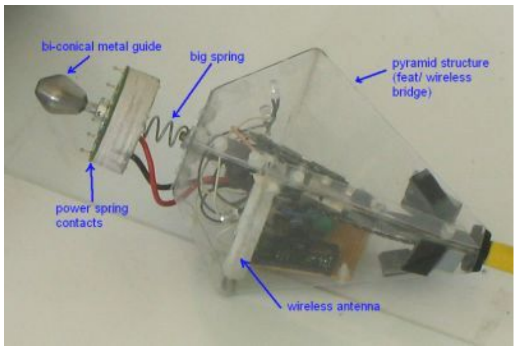

Fig. 1. Docking pyramid

In [6], the robot moves towards the mother robot but the marker is relatively large (larger areas are easier to detect) and it is always visible. In RAPOSA, the situation is more complex because the docking station enters the robot by the hole that is used by the back camera to look for it. Moreover, the optical center of the camera is not mounted at the height of the station and the robot has to lower its back part in order to dock. So, control in the y-axis is also done. It could not be found any other work that controls in $3D$ the robot to enter the docking station recurring only to vision.

C. Solution proposed

The goal is to start the autonomous docking algorithm from the moment that the pyramid is visible by the back camera. The algorithm ends with the fastening of the sliding doors. Due to the location of the camera and the insertion hole size, the useful Field of View (FOV) is approximately $25°$. To minimally add visible marks to aid the vision algorithm, the bi-conical metal guide was painted with orange, and some edges of the pyramid were painted with blue (Fig. 2). This allows to both detect the cable end, and to estimate the relative orientation of the pyramid to the robot.

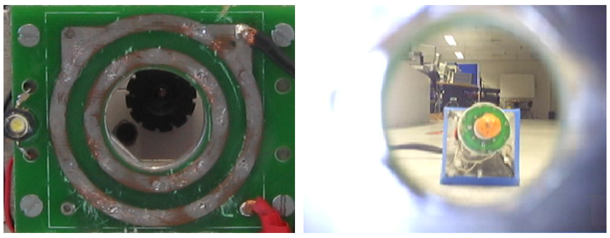

Fig. 2. Left: docking socket with the back camera visible inside the robot. Right: docking pyramid with the blue edges and the orange metal guide.

1) Docking regions: Fig. 3 depicts a top view of the geometry of the problem. The robot is assumed to have the pyramid within the FOV of its back camera. The relative angle α that the pyramid axis makes with the robot position is crucial to the docking process: when this angle is small, the robot is able to dock in a straightforward fashion, by going forward[1] while performing small adjustments. Otherwise, an alignment maneuver is necessary to bring it aligned with the pyramid axis, thus reducing α to a small value. The strategy followed to tackle this problem consists in dividing the space around the pyramid in angular sectors, relative to the pyramid axis, as shown in Fig. 4.

[1]In fact, the robot moves backwards relative to its front, since the robot docks its back with the docking station. We will, however, adhere to the word *forward* to avoid confusion.

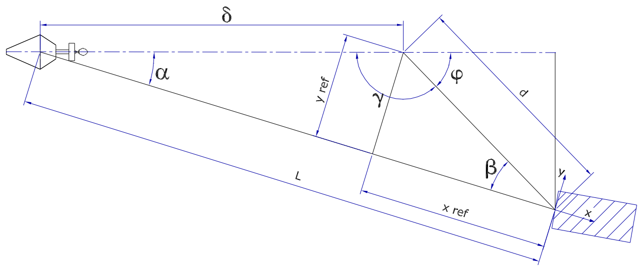

Fig. 3. Top view of the geometry of the problem: the horizontal axis is aligned with the pyramid, at the top left corner of the figure, while the robot is shown at the bottom right corner, with the pyramid in the FOV of the back camera.

Four regions were defined: region I, where the robot approaches the docking station going forward, while using visual servoing to make small adjustments to compensate for drifts in the alignment; region II, where the robot can detect both colors to estimate the angle α, thus performing a maneuver to place the robot within region I (and then use the region I strategy to perform the final approach); region III, where it is not possible for the robot to dock successfuly by lack of visual information, and a *deadzone* too close to the robot for it to operate without coliding with the pyramid. If $\alpha \geq 73°$ while in region II, the estimated α is incorrectly estimated, due to visual ambiguity, however it can be proved that the manoeuver performed in region II always re-positions the robot in region II with $\alpha < 73°$. The distinction between region I and II is not a straight line. A tolerance was introduced within a certain distance range, after preliminary experimentation, to accommodate for odometry errors that prevented the manoeuver in region II to conduct the robot to region I.

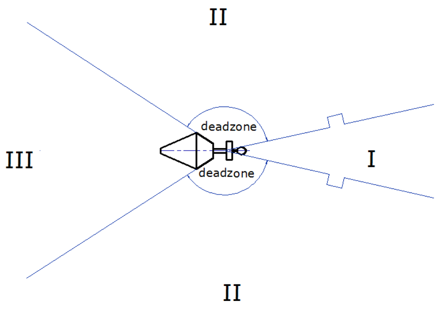

Fig. 4. The defined regions around the pyramid (top view).

2) Mathematical description: All the computation is based on the pin-hole model of the camera and on the perspective projection. The angle α is estimated from the detected blue rectangle (pyramid corners) and orange ellipse (ball at the pyramid end). The distance is estimated based on the geometric properties of the detected blue regions, whenever the robot is not aligned with the pyramid. Otherwise, it can use simply the orange ellipse area because the ellipse will maintain its shape as long as the robot moves in the direction of the pyramid. The blue color is not used here because occlusions occur for some distance range, while the orange is essential and enough for little misalignments. From the side view of the robot (Fig. 5), one can get by trigonometric relationships:

$$\frac{g}{f} = \frac{h'}{d}, \tag{1}$$

where f denotes the focal length of the camera, d is the distance between the focal point and the vertical passing through the top blue vertex of the pyramid, g is the height of the blue rectangle detected in pixels, while h' is an approximation of the length of the blue edge. While h is the real length of the edge, this approximation $h' \simeq h$ has a reasonable small error (max 1.5%). So, (1) is approximated here by

$$\frac{g}{f} \simeq \frac{h}{d}. \tag{2}$$

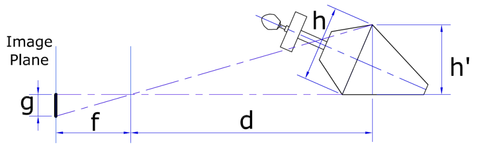

Fig. 5. Side view from the docking pyramid and its corresponding projection

From the top view (Fig. 6), the geometric relationship to estimate c is given by

$$\frac{c}{f} = \frac{a'+b'}{f} = \frac{a+b}{d'}. \tag{3}$$

However, since d' cannot be accurately estimated, we use the relation

$$\frac{c}{f} = \frac{a''+b''}{d} \simeq \frac{a+b''}{d}, \tag{4}$$

while assuming $a'' \simeq a$. The variables a and b'' are given by

$$a = m\sin(\alpha), \tag{5}$$

$$b'' = \frac{h}{2}\cos(\alpha), \tag{6}$$

where m is the distance between the midpoint of a blue edge and the center of the ellipse and α the angle to be estimated.

Using (4) and (2), we can obtain

$$\sin(\alpha) = \frac{chm}{g(m^2+(\frac{h}{2})^2)} \pm \frac{\sqrt{(\frac{ch}{g}m)^2-(m^2+(\frac{h}{2})^2)((\frac{ch}{g})^2-(\frac{h}{2})^2)}}{(m^2+(\frac{h}{2})^2)}. \tag{7}$$

As any quadratic equation, (7) has always 2 solutions (ignoring singularities). So, one has to chose one of them. If

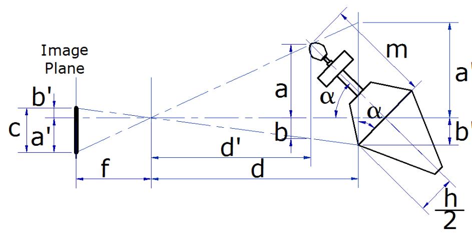

Fig. 6. Top view from the docking pyramid and its corresponding projection

one equals the term $(\frac{ch}{g})^2 - (\frac{h}{2})^2)$ to 0, $\frac{c}{g}$ is 0.5 meaning that the distance from the ellipse center to the edge end is half the width of the edge. This situation occurs at $0°$ so it implies $\sin(\alpha) = 0$ which requires choosing the minus sign on the solution of the quadratic equation. For continuity reasons, the solution used will be always with the minus sign. Even though, there are still 2 possible solutions. The inflexion point corresponds to about $73°$ which means that the angles estimated will be in the range $[0, 73]°$ (for both sides). For angles greater than that inflexion point there is an estimation error. Although, the robot will always get closer to the pyramid and will enter a position where $\alpha < 73°$ eliminating the error after one step for initial angles lower than $110°$ which is also the maximum angle for which the robot can see both colors without occlusions. When the robot is not aligned (non-small α), there is the need to align it. To do so, and taking in consideration the geometry formulation in Fig. 3, the robot rotates $-\beta$ at a first step (relative to referential where the robot is looking straightforward to the pyramid), travels the distance d and then rotates φ. This will position the robot at a distance δ of the pyramid aligned with it.

3) Vision algorithm: The high level vision algorithm could be generically described in 3 steps:

- color based image segmentation
- shape based landmark extraction
- landmark recognition

For the color based segmentation, the color space chosen was HSV (Hue, Saturation, Value). A color is specified by a volume in this space, defined by the intersection of one interval for each axis H, S, and V. These slices offer a more robust detection performance, since RGB is very sensitive to lighting [12]. Still, lighting changes prevent the use of fixed and unchangeable thresholds to define the bands of H, S and V components. So, it is given to the user the possibility of adjusting the thresholds in real time. The use of HSV allows for an intuitive tuning, contrasting with adjustment in RGB that could be fastidious.

For the shape based analysis, several metrics are used both for elimination and for choosing the best candidate among several ones. For the orange color there is an ellipse fitting for all the orange blobs and then some ellipses are filtered out first by area and then by ratio between axes. For the blue color, the process is similar. But here, the fitting of the rectangle is not by the circumscribed rectangle of minimal area but by its bounding region. This is so because using the bounding region is better in the presence of a bad detection (or occlusions) as shown in Fig. 7.

After this, other geometric constrains are used to eliminate pairs of rectangles and ellipses that cannot correspond to the physical dimensions of the object. There is a maximum deviation both for height and width scaled to the detection distance. There is also an area ratio criterion between the ellipse and the rectangle. A tracking of the ellipse is done: any pair with an ellipse out of a ball of 20 pixels (configurable) around the previous ellipse is ignored. After all these filters, if there is more than one possible pair, the same criteria are used but this time tighter. In this way, one is more sensible to noise but in the case of a bad detection where there is only one pair, there is no risk of eliminating it by too strict constrains. Fig. 7 is an example of a well chosen candidate among 3 possible ones.

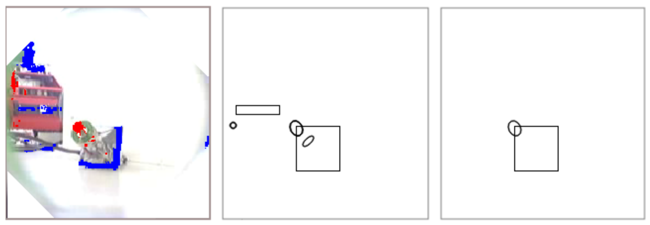
Fig. 7. On the left, the original image. In the middle, the candidate pairs after the filtering operation. On the right, the chosen candidate by the metrics explained in the main text.

After detecting the best orange and blue candidates regions, one of 7 states is chosen. Each state determines a specific course of action, as described in the next section. The choice of the state is performed as follows. After choosing the best pair, the state is chosen between 1 or 2 according to the angle and distance. If the robot is too close to the pyramid, the estimation of the angle is not accurate and then state 1 is chosen. That implies the *deadzone* mentioned. For angles bigger than $15°$ and/or distances bigger than $34cm$, state 2 is chosen. If there is no pair at the end of the detection, then the state is chosen by the orange ellipses detected at a first option. The best ellipse is chosen (by the same criteria) and depending on the distance, state 1, 3 (big distance) or 6 (small distance) is chosen. In the case of no detected ellipse, if there is any blue rectangle state 4 is chosen, otherwise state 5 is the one used. When the robot is at either states 6 or 7, further state choices are inhibited, *i.e.,* it remains in that state until the corresponding behaviour finishes (namely, the completion of a docking or a rotation). The end of state 7 is detected based on a predictor to prevent the loss of the ellipse (in the case that the robot has to rotate all the FOV) due to the delay on the control cycle. The predictor uses the difference between the present and the previous frame to predict where the robot will be in the next one.

D. Motor schema

For each state, the velocities are computed in different ways. The arm position is also controlled but only at the end of state 1 near the docking pyramid. In all other states, the arm is kept at a constant position.

For region I, states 1 and 6 are used. In state 1, the linear velocity is constant when the robot is above a certain distance ($\simeq 1m$). Below that distance, linear velocity is given by $K_{linear}\sqrt{A_{ellipse}}$, where $A_{ellipse}$ is the ratio between the ellipse area at 1m (offline reference image) and the ellipse area at the current frame, and K_{linear} is a gain. The square root is used to make the velocity approximately linear with the distance, because the area of an ellipse decreases quadratically with the distance. The angular velocity is given by $-K_{angular}\, x_{deviation}$, where $K_{angular}$ is a gain, while $x_{deviation}$ is the shift in pixels in the x-axis from the image center.

The arm position is set at the end of state 1 to $-K_{arm}\, y_{deviation}$, where K_{arm} is a gain, while $y_{deviation}$ is the shift in pixels in the y-axis from the image center. At the same time that the arm starts to move, the lights are turned on. This is because the state 6 is used when the area detected is higher than a certain threshold (correspondent to $0.3m$). In this situation the lights help the detection nearby. In this final state (6), the velocity is kept constant at a low value until a certain amount of time. After this, the robot stops and starts closing the bi-conical metal guides (see section II-B).

For region II, RAPOSA cannot keep visual contact with the pyramid at all times, and thus it has to rely on odometry (see section II-E). Odometry is used only whenever the pyramid is out of the robot FOV. Visual odometry is not advised due to the low quality of the images.

The rotation based on vision corresponds to state 7. The first step of the algorithm for region II is a pure rotation of $-\beta$. Whenever the angle $-\beta$ lies within the FOV of the image, the first step in region II will use only state 7, using visual servoing alone. When it is not, the angle to rotate is divided in two segments: the first part is limited by the FOV (state 7) and the second part is an odometry based rotation. The velocities used in state 7 have a low constant value so it can track the ellipse during the rotation. But, if the robot looses the ellipse for some moments, it stops. If the number of sequenced frames without identifying again the ellipse is too high, it exits state 7 and recomputes a new state.

To reduce the odometry error, an odometric calibration is done, where the origin is reset after the end of state 7 with the angle rotated in that state. The inverse controller technique is used then in state 2. The velocities resulted from the controller are saturated and scaled, and different tolerances are used for each step mentioned at the end of subsubsection II-C.2. The x_{ref} and y_{ref} of Fig. 3 are the references for traveling distance d. The use of a bi-dimensional controller is justified by the possibility of correcting some orientation error in the middle step of going straight distance d. The last rotation is not done based on vision due to the blurring noise. It is preferable to rotate based on odometry and then, after the robot is stopped, if it does not identify the pyramid, rotate slowly until it finds it again.

This rotation is not performed always to the same side, but rather to the side opposite to the one in the previous rotation.

Moreover, instead of endlessly rotating if no landmark is found, the robot switches direction after a certain angle lower than $360°$. While no landmark is found, this angle is doubled after each switch, so that is starts scanning a small angular range, and increases this range progressively.

E. Odometry

Odometry was not already implemented in the robot, so there was the need to develop it. It is extremely unadvised in the literature to use dead reckoning in this kind of tracked vehicles [13], [14]. Although RAPOSA is a differential drive vehicle, it is a particular case of it named skid steering. In skid-steer vehicles, there is always a large slippage whenever the vehicle turns. In RAPOSA, the center of rotation also changes, depending on the position of the frontal arm. One possible solution to estimate a more accurate kinematic model could be to use the approach described in [15], but it would need additional external sensors that are not feasible to introduce in RAPOSA.

Considering the classic unicycle kinematic model (Fig. 8), the equations relating the vehicle velocity and the speed of each wheel are the following:

$$V = h\frac{\omega_r + \omega_l}{2} = \frac{V_r + V_l}{2} \quad (8)$$

$$\omega = h\frac{\omega_r - \omega_l}{d} = \frac{V_r - V_l}{d} \quad (9)$$

where V and ω are respectively the linear and angular velocities, V_r and V_l denote the right and left wheel speeds, h is the wheel radius (assumed equal to both wheels), and d is the distance between the wheels. As it is described in [13], the distance d, in the case of a tracked wheel vehicle, lies within an interval between d_{min} and d_{max} (see Fig. 9). This distance is not actually fixed, depending for instance on the ground physical properties. However, in this implementation it was empirically estimated and considered a constant.

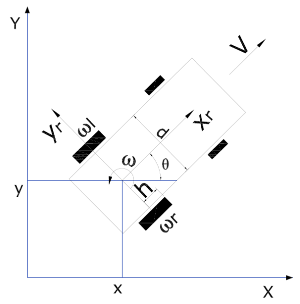

Fig. 8. Kinematic model of a unicycle vehicle.

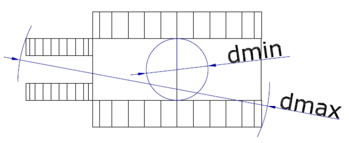

Fig. 9. Range of the d distance for the case of RAPOSA.

Low level issues, such as non-accurate access to each wheel velocity, together with an analog drift introduced a significant error margin to the odometry measurement.

F. Results

In order to compare the results of this work with others, the benchmark used in [6] and shown in Fig. 10 was chosen. These points are the starting points for the robot and in each point the robot is pointing towards the docking station.

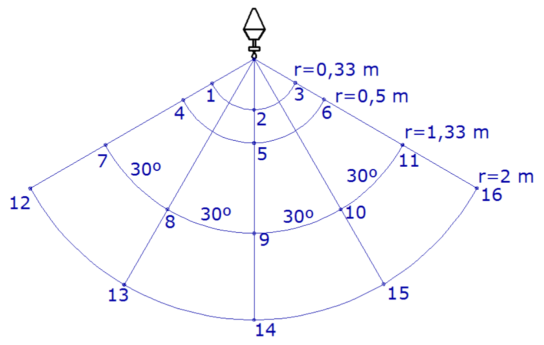

Fig. 10. The grid of points used in both works.

However, this grid of points is not completely comparable. Their robot had a FOV of $70°$ but for reliability reasons they defined an area with a maximum of $60°$(for each side). RAPOSA's FOV is $25°$ so using a region of about $120°$ is a more demanding challenge. The results for this grid of points are presented in Table I, showing the mean time to dock, the mean velocity (v_{mean}), the standard deviation (σ), and the percentage of successful dockings ($\%_{success}$). The number of trials was 4 for all the points.

Points 1 and 3 are not presented here because the robot was not able to dock in any of the trials. These points are inside the deadzone discussed previously, and so this was expected to occur. For point 4 it was not possible to dock in one of the trials.

It is clear that for the $0°$ angle the standard deviation is kept very low (less than $2.5s$) but the mean velocity increases. This can be understood under the light of the motors schema presented: recall that the velocity is linearly proportional to the distance except when it is too far or when it is too close, where it is kept constant at a moderate speed

TABLE I
RESULTS AS A FUNCTION OF THE ANGLE.

Angle (°)	Points	Mean Time to Dock (s)	σ (s)	v_{mean} (cm/s)	$\%_{success}$
0	2	19.41	2.04	1.7	100
	5	24.29	2.41	2.1	100
	9	37.88	2.04	3.5	100
	14	46.06	2.48	4.3	100
30	8,10	45.08	3.98	3.0	100
	13,15	51.10	5.24	3.9	100
60	4,6	92.77	25.18	0.5	87.5
	7,11	73.00	30.72	1.82	100
	12,16	68.34	12.18	2.9	100

or very low speed, respectively. This fact explains the results, since for points 9 and 14 the robot will move at a constant speed in the beginning, decreasing its speed only after a while. In point 2 the weight of the low speed mode is greater, thus explaining a lower mean velocity. For the $30°$ angle, the situation is the same for the mean velocity (increasing here σ). The reasons here are different, since for a higher distance, the movement is softer and more reliable because vision is used longer, and the generated trajectory has smaller rotations. Using vision produces less error at the end of state 2, which contributes to a faster convergence towards region I. Moreover, for a greater distance, the angle estimation error is smaller because the approximations used have smaller errors. For the $60°$ angle, the analysis for the mean velocity is the same than for the $30°$ angle. The standard deviation is much larger than before even though at $2m$ is significantly lower than for the other distances. Softer movements and low estimation error explain that fact. The worst situation occurs for points 4 and 6 in terms of mean velocity. The lower v_{mean} makes sense, after considering the fact that the robot has to move backwards and that increases the traveled distance.

The grid of points used in the experiments above could not explore all the possibilities of the algorithm, including the recovery of the estimation error in one step for angles lower than about $110°$. So, other points were tested including angles of $80°$, $90°$ and $110°$. For all these points, the robot successfully although sometimes with too much time ranging from $1min$ till almost $4min$ for distances of about $1.5m$. The minimum iterations on region II was 2 as expected and the maximum was 4. The problem was not with the angle estimation after the first iteration. The problems are mostly related with the odometry and with the noise in states 3, 4 and 5. Remember that when there is an estimation error, the robot might not be aligned after the first iteration, and might enter one of those states (3, 4 or 5). The robot could manage to identify correctly the pyramid but sometimes took longer because it entered state 3 or 4 with other objects before finding the pyramid. The maximum distance was also tested at $0°$ angle. After adding a point at $2.5m$ with 100% of docking success, other distances were tested. At $2.7m$ the robot performed well, at $2.8m$ it docked successfully only in 75% of the trials, and at $3m$ it does not dock at all. At $3m$ it is not possible to extract enough information from the image received. For other angles, the maximum distance is the $2m$ radius discussed previously. Finally, the robustness to experimental conditions was tested and the robot was able to dock in an outside environment even though the odometry was not corrected for that soil. Fig. 11 shows a very noisy example for which the robot docked. Even for an angle different of $0°$ (about $25°$) it managed to dock in less than $80s$ even though there are big differences in the terrain.

III. Conclusion and Future Work

An autonomous docking algorithm for a SAR robot was presented in this paper. This mechanism employs vision most of the time, recurring to the odometry only when deemed necessary. Results have shown a good performance. Future work includes improvement of the odometry, which being a tracked wheels robot it becomes a non-trivial task, as well as improvement of the color segmentation, so that the need to adjust manually the HSV bands thresholds, e.g. because of lighting conditions, can be eliminated.

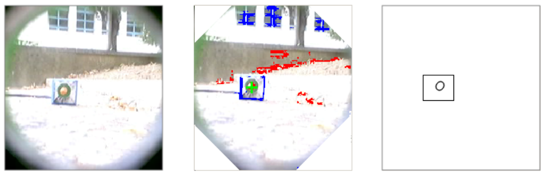
Fig. 11. An example of the color detection and segmentation in outdoor environment.

References

[1] R. Murphy, J. Casper, J. Hyams, M. Micire, and B. Minten, "Mobility and sensing demands in usar," in *IECON-2000: 26th Annual Conference of the IEEE*, vol. 1. Industrial Electronics Society, 2000, pp. 138–142.

[2] C. Marques, J. Cristvo, P. Lima, J. Frazo, I. Ribeiro, and R. Ventura, "Raposa: Semi-autonomous robot for rescue operations," *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 3988–3993, Oct. 2006.

[3] W. G. Walter, *The Living Brain*. New York: W. W. Norton, 1953.

[4] K. T. Seungjun Oh, Alexander Zelinsky, "Autonomous battery recharging for indoor mobile robots," in *Proceedings of Australian Conference on Robotics and Automation (ACRA2000)*, 2000.

[5] M. Kim, H. W. Kim, and N. Y. Chong, "Automated robot docking using direction sensing rfid," *2007 IEEE International Conference on Robotics and Automation (ICRA 2007). Proceedings.*, pp. 4588–4593, April 2007.

[6] B. Minten, R. Murphy, J. Hyams, and M. Micire, "Low-order-complexity vision-based docking," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 922–930, December 2001.

[7] M. C. Silverman, D. Nies, B. Jung, and G. S. Sukhatme, "Staying alive: a docking station for autonomous robot recharging," *IEEE International Conference on Robotics and Automation, 2002. (ICRA '02). Proceedings.*, vol. 1, pp. 1050–1055 vol.1, May 2002.

[8] R. Wei, R. Mahony, and D. Austin, "A bearing-only control law for stable docking of unicycles," *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings.*, vol. 4, pp. 3793–3798 vol.3, Oct. 2003.

[9] R. Luo, C. Liao, K. Su, and K. Lin, "Automatic docking and recharging system for autonomous security robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005. (IROS 2005).*, pp. 2953–2958, Aug. 2005.

[10] U. Kartoun, H. Stern, Y. Edan, C. Feied, J. Handler, M. Smith, and M. Gillam, "Vision-based autonomous robot self-docking and recharging," *World Automation Congress, 2006. WAC '06*, pp. 1–8, July 2006.

[11] C. Marques, J. Cristovão, P. Alvito, P. Lima, J. Frazão, M. I. Ribeiro, and R. Ventura, "A search and rescue robot with tele-operated tether docking system," *Industrial Robot*, vol. 34, no. 4, pp. 332–338, 2007.

[12] V. L. B. Bascle, O. Bernier, "Learning invariants to illumination changes typical of indoor environments: Application to image color correction," *International Journal of Imaging Systems and Technology*, vol. 17, no. Issue 3, pp. 132–142, Oct. 2007.

[13] H. R. Everett, *Sensors for mobile robots: theory and application*. Natick, MA, USA: A. K. Peters, Ltd., 1995.

[14] R. Siegwart and I. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Bradford Book, 2004.

[15] J. Martinez, A. Mandow, J. Morales, A. Garcia, and S. Pedraza, "Kinematic modelling of tracked vehicles by experimental identification," *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings.*, vol. 2, pp. 1487–1492, Oct 2004.